Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 6, June 2015, pg.157 – 164

RESEARCH ARTICLE

EVALUATING THE DIVERSE ALGORITHMS OF TRANSMISSION CONTROL PROTOCOL UNDER THE ENVIRONMENT OF NS-2

Ravi Kamboj, *M.Tech Student, Department of Computer Science and Engineering, Yamuna Institute of Engineering and Technology, Gadholi.*

Gurpreet Singh*, Dean Academics, Associate Professor, Department of Computer Science and Engineering, Yamuna Institute of Engineering and Technology, Gadholi.*

ABSTRACT-*The purpose of this paper is to analyze and compare the different congestion control and avoidance mechanisms which have been proposed for TCP/IP protocols. In this paper, we analyze and compare the performance metrics of different TCP variants, namely TCP Fack, TCP Sack and Rate Based Pacing to classify which variant of TCP performs better in terms of Throughput, Jitter, End to End Delay, Delivery ratio and how many numbers of packet loss. A table is also drawn which shows the comparison results of the variants. This analysis will be useful in determining the best variant among TCP Protocols to ensure better data transfer, speed, and reliability and congestion control. TCP congestion control has been designed to ensure internet stability along with fair and efficient allocation of network bandwidth. To overcome the congestion problem several congestion control and avoiding mechanisms, namely TCP Fack, TCP Sack, Rate Based Pacing are discussed in this paper.*

KEYWORD-*Transmission Control Protocol (TCP), Sack, Fack, Rate Based Pacing.*

I. INTRODUCTION

The TCP is a reliable connection-oriented stream protocol in the Internet Protocol suite [2]. A TCP connection is like a virtual circuit between two computers, conceptually very much like a telephone connection. To maintain this virtual circuit, TCP at each end needs to store information on the current status of the connection e. g. The last byte sent.

TCP is called a connection-oriented because it stands with a 3-way handshake protocol and it maintains its state information for each connection [7].TCP is also called a stream protocol.TCP guarantees that it will deliver data supplied by the application to the other end, although the underlying Layers (IP) offer only unreliable data delivery i.e, even if the data gets lost, corrupted or duplicated. Sometimes the data transmit from one end to the other or is delivered out of order by the communication system, then TCP is responsible for delivering error-free data in-order to the application at the other end. Hence, it is reliable. This reliability is achieved by assigning a sequence number to each byte of data that is transmitted [13]. It is also required by the receiving TCP to send positive acknowledgments (ACKs) back to the data sender. This acknowledgment mentions the next byte of data expected by the receiver.

TCP sources break messages from higher protocol layers into datagram's that are encapsulated in packets which are then transmitted over the network. These packets are reassembled by the TCP receiver into the original message and passed on to the higher level protocol layers. For every packet sent on the network by a source an acknowledgement (ACK)[4] is expected to be transmitted back from the destination. This ACK is used by the source to determine if the transmitted packet was successfully received at the destination or not. In this manner packets can be tracked and retransmitted if required.

II. CONGESTION AVOIDENCE

A network is considered congested when too many packets try to access the same router's buffer, resulting in an amount of packets being dropped. In this state, the load exceeds the network capacity [15]. During congestion, actions need to be taken by both the transmission protocols and the network routers in order to avoid a congestive collapse and furthermore to ensure network stability, throughput efficiency and fair resource allocation to network users. Indeed, during a collapse, only a fraction of the existing bandwidth is utilized by traffic that finally reaches the receiver.

Congestion is considered [3], in general, as a catastrophic event. However, congestion itself is associated with different properties, depending on the characteristics of the underlying networks, the mechanisms of the transmission protocols, the traffic characteristics of the contenting flows, the level of flow contention, and the functionality of network routers.

During the initial data transfer phase of a TCP connection the Slow Start algorithm is used. However, there may be a point during Slow Start that the network is forced to drop one or more packets due to overload or congestion [8][11]. If this happens, Congestion Avoidance is used to slow the transmission rate. However, Slow Start is used in conjunction with Congestion Avoidance as the means to get the data transfer going again so it doesn't slow down and stay slow.

In the Congestion Avoidance algorithm [4] a retransmission timer expiring or the reception of duplicate ACKs can implicitly signal the sender that a network congestion situation is occurring. The sender immediately sets its transmission window to one half of the current window size (the minimum of the congestion window and the receiver's advertised window size), but to at least two segments. If congestion was indicated by a timeout, the congestion window is reset to one segment, which automatically puts the sender into Slow Start mode. If congestion was indicated by duplicate ACKs, the Fast Retransmit and Fast Recovery algorithms are invoked [8]. As data are received during Congestion Avoidance, the congestion window is increased. However, Slow Start is only used up to the halfway point where congestion originally occurred. This halfway point was recorded earlier as the new transmission window. After this halfway point, the congestion window is increased by one segment for all segments in the transmission window that are acknowledged. This mechanism will force the sender to more slowly grow its transmission rate, as it will approach the point where congestion had previously been detected [12].

III. FEW VARIANTS OF TCP

There are numbers of variants of TCP, named as:-

a. TCP Sack
b. TCP Fack
c. RBP

a. TCP SACK (Selective Acknowledgments)

The congestion control algorithms implemented in our Sack TCP are a conservative extension of Reno's congestion control [9], in that they use the same algorithms for increasing and decreasing the congestion window, and make minimal changes in the other congestion control algorithms. Adding Sack to TCP does not change the basic underlying congestion control algorithms. The Sack TCP implementation preserves the properties of Tahoe and Reno TCP of being robust in the presence of out-of-order packets, and Uses retransmit timeouts as the recovery method of last resort. The main difference between the Sack TCP implementation and the Reno TCP implementation is in the behavior when multiple packets are dropped from one window of data. As in Reno, the Sack TCP implementation enters Fast Recovery when the data sender receives TCP retransmit-thresh duplicate acknowledgments. The sender re-transmits a packet, and cuts the congestion window in half.

During Fast Recovery [6], Sack maintains a variable called pipe that represents the estimated number of packets outstanding in the path. (This differs from the mechanisms in the Reno implementation). The sender only sends new or retransmitted data when the estimated number of packets in the path is less than the congestion window. The variable pipe is incremented by one when the sender either sends a new packet or retransmits an old packet. It is decremented by one when the sender receives a dup ACK packet with a Sack option reporting that new data has been received at the receiver. Use of the pipe variable decouples the decision of when to send a packet from the decision of which packet to send. The sender maintains a data structure, the scoreboard that remembers acknowledgments from previous Sack options. When the sender is allowed to send a packet, it retransmits the next packet from the list of packets inferred to be missing at the receiver. If there are no such packets and the receiver's advertised window is sufficiently large, the sender sends a new packet. When a retransmitted packet is itself dropped, the Sack implementation detects the drop with a retransmit timeout, retransmitting the dropped packet and then slow-starting. The sender exits Fast Recovery when a recovery acknowledgment is received acknowledging all data that was outstanding when Fast Recovery was entered.

Problems:

The biggest problem with Sack is that currently selective acknowledgments are not provided by the receiver. To implement Sack we'll need to implement selective acknowledgment which is not a very easy task.

b. TCP FACK (Forward Acknowledgments)

Forward Acknowledgments (FACK) also aims at better recovery from multiple losses [3]. The name "Forward ACKs" comes from the fact that the algorithm keeps track of the correctly received data with the highest sequence number. The name "forward ACKs" comes from the fact that the algorithm keeps track of the correctly received data with the highest sequence number [3]. In Fack, TCP maintains 2 additional variables, Fack, that represents the forward most segments that has been acknowledged by the receiver through the Sack option, and second return data, that reflects the amount of outstanding retransmitted data in the network. Using these two variants, the amount of outstanding data during recovery can be estimated as forward-most data sent and forward most data Acknowledged is the submission of Fack value and outstanding retransmitted data. TCP Fack regulates this value (the amount of outstanding data in the network) to be within one segment of cwnd, cwnd remains constant during the fast recovery period. The fack variable is also used to trigger the fast retransmit more promptly [14].

c. RATE BASED PACING (RBP)

Prior work has suggested that this “slow-start restart" problem is a contributor to poor performance of P-HTTP over TCP. One way of solving the problem is to send segments at a certain pace until we get the ACK clock running again [6]. This pace or rate should be based on a fraction of prior estimates of data transfer rate, since that is the closest estimate of available bandwidth that we have (if we had some magical way of knowing the exact available bandwidth at the end of the idle time, we could have used that). We believe that this modification, called Rate Based Pacing (RBP), will give better performance for the circumstances mentioned in the problem.

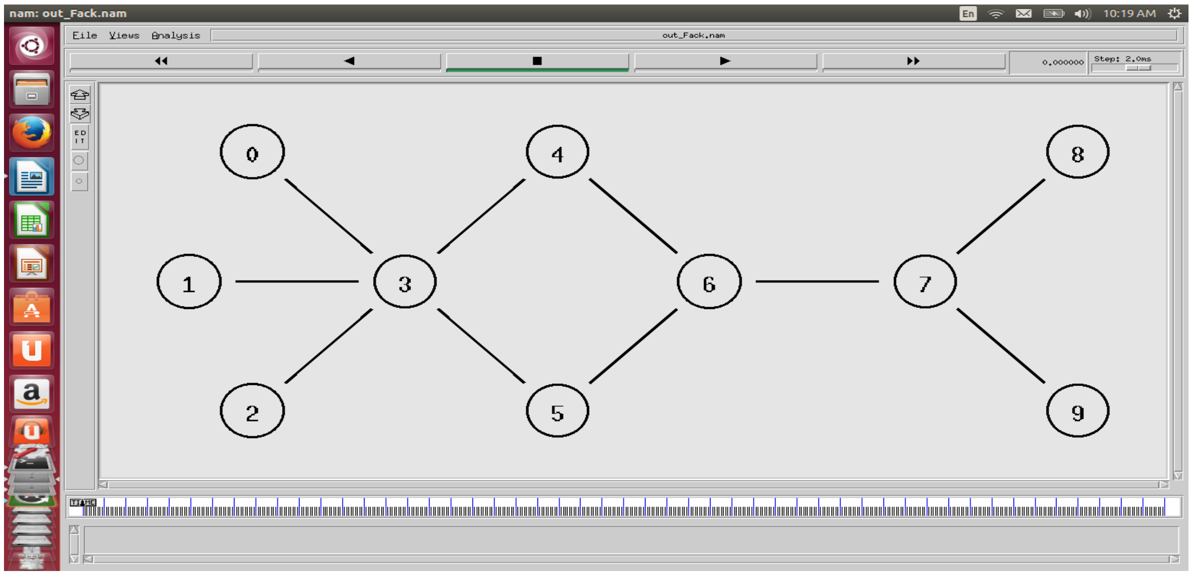
(i) RBP IMPLEMENTATION

Rate based pacing requires the following changes to TCP:

1. Bandwidth estimation.
2. Calculation of the window that we expect to send in RBP and the timing between segments in that window.
3. A mechanism that clocks the segments sent in RBP.

IV. SIMULATION AND RESULTES

This paper shows a wired topology of ten nodes represented in the figure nodes labeled as node 0,node 1 and node 2 are connected with node 3 via a duplex link and node 3 is connected with node 4 and node 5 with two way open communication channel and node 4 and node 5 are also connected with node 6 via duplex link, and node 6 is connected with node 7 and node 7 is connected with node 8 and node 9 with duplex link.TCP agents of different variations are attached with node n0(source node) and send the FTP data to the corresponding destination node 8 and similarly node 2 is the source node for the node 9 and send the packets to the destination node. We discussed the results of the simulated scenario of TCP variants as TCP Sack, TCP Fack and Rate Based Pacing, and compare these results. After obtaining the results we draw a graph to display the performance of variants according to the five parameters.



s

Figure 1: Topology for implementation of few variants of TCP.

We discussed the results of the simulated scenario of TCP variants Sack, Fack, Rate Based Pacing with different parameters using the Ns-2 simulator.

In this paper, we consider the different parameters of TCP variants as Throughput, No of packet loss, Delivery ratio, End to End delay and Jitter. We have done a practical on the given topology in the paper and compare the results of various TCP variants.

On the basis of these parameters which we included in the practical, the values generated in the experiment are given below in the tabular form.We build various graphs on the basis of these values and find out the best variants.

Table 1:Tabular Representation of TCP Variants:

	TCP Fack	TCP Sack	RBP
Throughput	4004.255	4349.895	4678.064
No.of packet loss	51	73	0
Delivery Ratio	99.828	99.751	99.949
End to End Delay	2328.546	2257.763	1459.354
Jitter	1504.641	3565.299	404.132

(i) ***Throughput:*** Throughput is defined as the total amount of data received by destination node from the source node divided by the total time it takes for the destination to get the last packet and it measures is packets per second.

Throughput=Node Throughputs of data transmission/Total Number of Nodes.

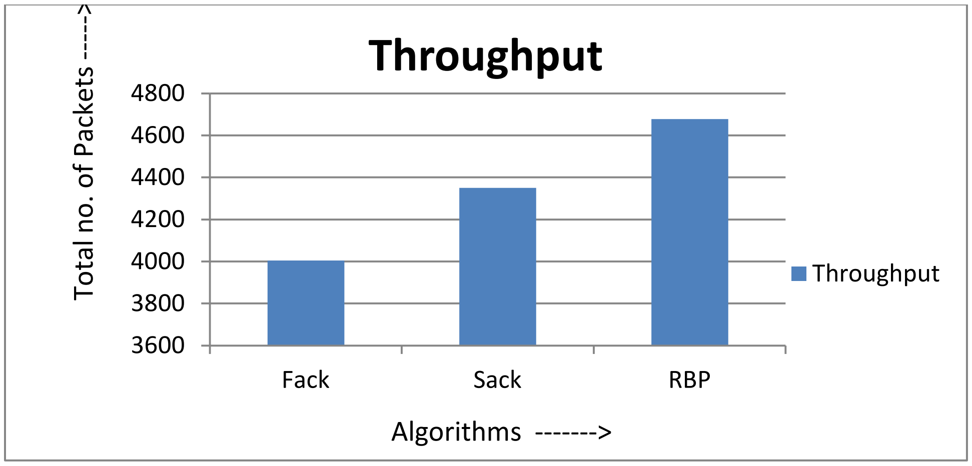


Figure: 2 Throughput

Fig 2. Shows the throughput in terms of bytes. This shows that RBP has better throughput as compared to others has low throughput value.

(ii) ***Number of packet Loss:*** Failure of one or more transmitted packets to arrive at their destination. Known as Packet loss.

No of packet loss=No of packets sent - No of packet Received

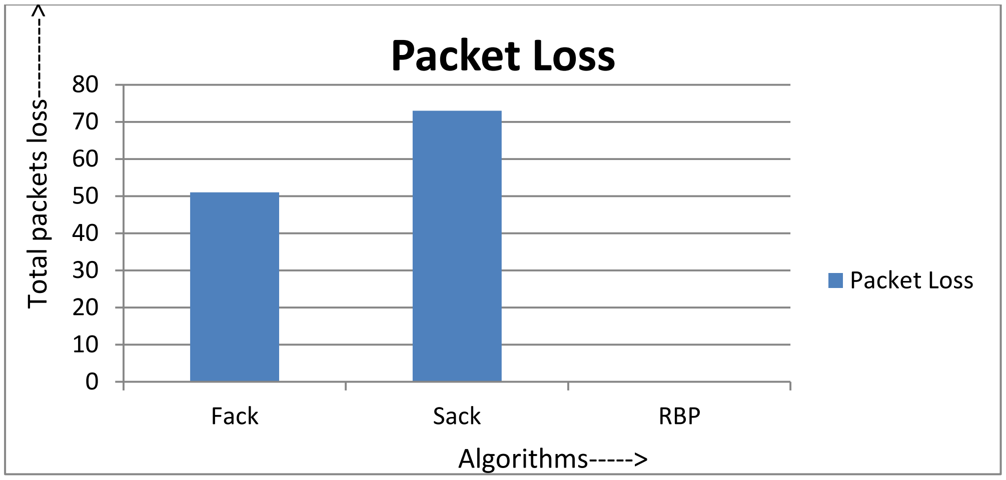


Figure: 3 packet loss

Fig 3. shows the number of packet loss in terms of bytes. This shows that the data rate of RBP is better than other variants. None of the packet was lost in RBP for the taken simulated topology.

(iii) ***Delivery Ratio****:* In TCP the Packet delivery ratio is the ratio of total packets sent by the source node to the successfully received packets by the destination node.

Delivery Ratio= Total data packet delivered successfully x 100 /data packet generated

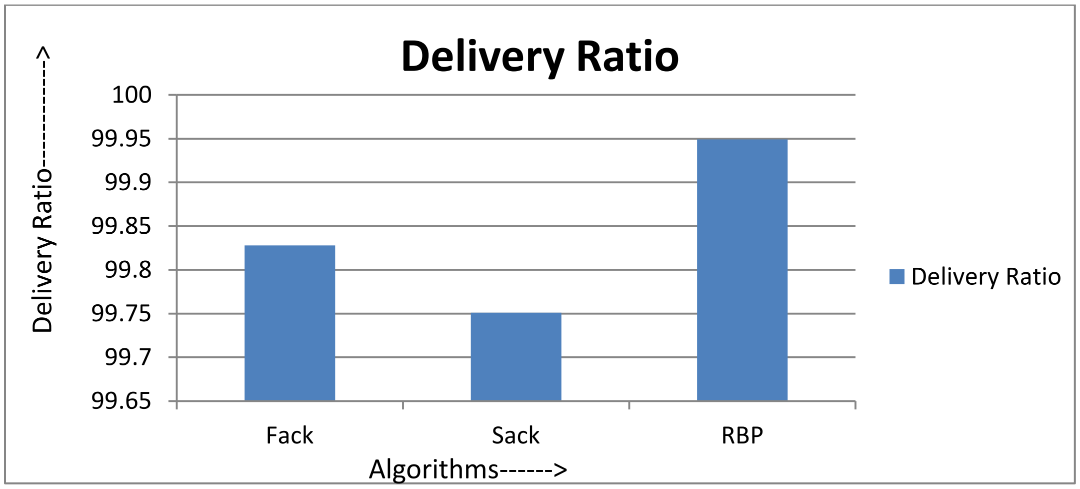


Figure:4 Delivery Ratio

Fig 4. Shows the delivery ratio of TCP in terms of bytes. This shows that RBP has a better delivery ratio than other variants.

(iv) ***Jitter:*** The time difference in packet inter-arrival time to their destination can be called **jitter**. Jitter is a specific issue that normally exists in the packet.

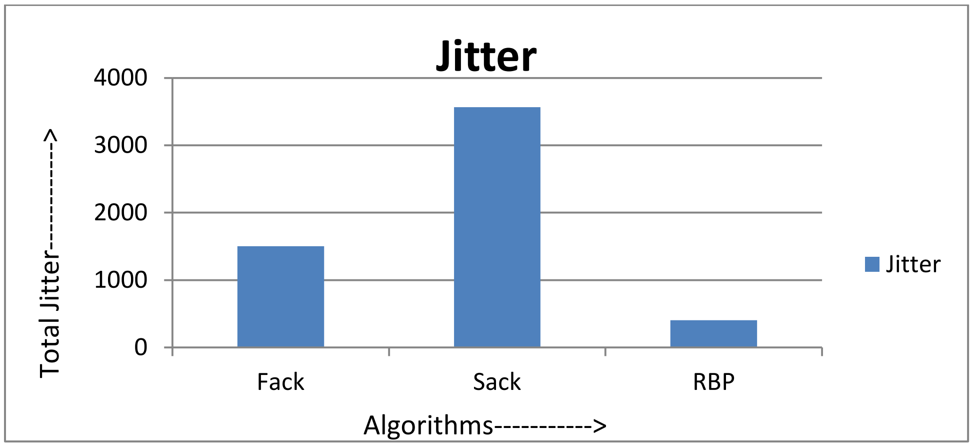


Figure: 5 Jitter

Fig 5. Shows the time difference in inter arrival time to their destination. RBP has lowest inter arrival time among all the three TCP variants which we discussed here.

(v) ***Average Delay:*** Average end-to-end delay is the time interval when a data packet generated from source node is completely received to the destination node.

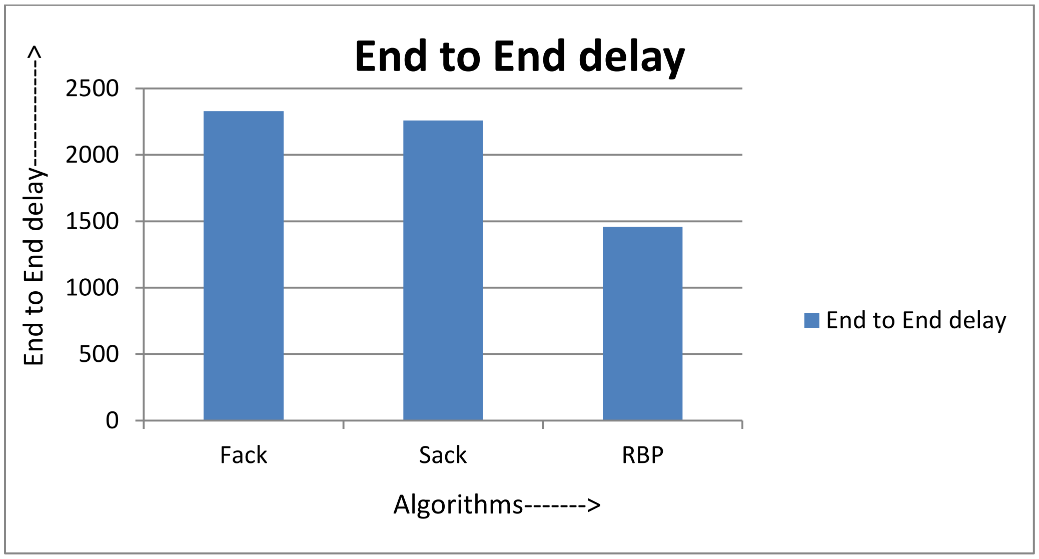


Figure: 6 End to End delay

Fig 6. Shows the End to End delay of all the TCP variants. This shows that the RBP has a less End to End delay as compared to other variants.

CONCLUSION

This paper compares the TCP variants performance using the NS2 simulator with different parameters Throughput, Number of packet loss, Jitter, Delivery Ratio, End to End Delay. Simulation results shown through graphs represent overall performances of TCP variants. From the obtained results shown by graphs, we can say that RBP shows highest efficiency and performs best in our simulated topology. This can also be verified from the data represented through table 1 and various figures from figure 2 to figure 6.

REFERENCES

1. Neha Bhatla, Amanpreet Kaur, Gurpreet Singh, “Congestion Control Techniques in TCP: A Critique”, in the proceedings of 3rd National Conference of Advances and Research in Technology (ART-2014), Pages 45.1-45.5, 8-9 March, 2014.
2. Mathis and J. Mahdavi. Forward acknowledgement: refining TCP congestion control. SIGCOMM Computer Commun. Rev26(4): 281–291, Aug. 1996.
3. J. Nagle Congestion Control in IP/TCP Internetworks. RFC 896, Jan. 1984.
4. L.S. Brakmo, S. W. O’Malley, and L. L. Peterson.Tcp Vegas: new techniques for congestion detection and avoidance. SIGCOMM Comput. Communication. Rev., 24(4):24–35, 1994.
5. W. Richard Stevens. TCP/IP Illustrated, Volume 1: The Protocols. Addison Wesley, 1994.
6. Amanpreet Kaur, Gurpreet Singh, Baninder Singh, “Evaluation of Congestion control variants of TCP by strolling propagation delay in NS-2”, International Journal of Applied Engineering and Research, Volume V, April, 2011
7. M. Allman, V. Paxson, and W. Stevens. TCP congestion control. RFC 2581, April 1999.
8. ACM Computer Communication Review, volume 26, pages 5-21, July 1996.
9. K. Leung and Vivtor O.K. Li, “Transmission Control Protocol (TCP) in wireless Networks: issues, approaches and challenges,” IEEE Communications Survey, 8 (4), pp. 64-79, 4th October 2006.
10. D.D.Clark, “Window and Acknowledgement Strategy in TCP”, RFC 813, July 1982.
11. Shalu Sraw, Gurpreet Singh, Amanpreet Kaur, “A Survey of Multicast Routing Protocols in MANETS”, In the proceedings 2nd International Conference on Futuristic Trends in Engineering & Management 2014 (ICFTEM-2014), pages 220-224, 3-4 May 2014.
12. L.S.Brakmo, L.L. Peterson, “TCP Vegas: End to End Congestion Avoidance on a Global Internet”, IEEE Journal on Selected Areas in Communication, vol.13, pages (1465-1490), 1995.
13. S. Floyd and V. Jacobson. “Random early detection gateways for congestion avoidance”, IEEE-ACM Transactions of Networking, Volume 1, Issuse 4, Pages 397-413, August 1996.
14. K. Fall and S. Floyd. “Simulation-based comparisons of Tahoe, Reno, and SACK TCP”, In ACM Computer Communication Review, volume 26, pages 5-21, July 1996.
15. Yuvaraju B., N. Niranjan, N Chiplunkar “Scenario Based Performance Analysis of Variants of TCP using NS2-Simulator” International Journal of Advancements in Technology, 1 (2), October 2010.
16. Amanpreet Kaur, Gurpreet Singh, Baninder Singh, “Evaluation of Congestion control variants of TCP by strolling propagation delay in NS-2”, International Journal for Applied Engineering and Research, 5, pp. 655-658, April, 2011.
17. S. Floyd and V. Jacobson, Random early detection gateways for congestion avoidance. /EEElACM Transactions Of! Networking, 1(4):397-413, August 1993.
18. Ravi Kamboj, Gurpreet Singh, “Various TCP Options for Congestion Evasion”, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 4, Issue 4, Pages 1534-1539, April 2015.